COMBINATORICA
Akadémiai Kiadó – Springer-Verlag

COMBINATORICA 11 (2) (1991) 97–122

PARALLEL COMPARISON ALGORITHMS FOR APPROXIMATION PROBLEMS

N. ALON* and Y. AZAR

Received August 22, 1988

Suppose we have n elements from a totally ordered domain, and we are allowed to perform p parallel comparisons in each time unit (= round). In this paper we determine, up to a constant factor, the time complexity of several approximation problems in the common parallel comparison tree model of Valiant, for all admissible values of n, p and ε, where ε is an accuracy parameter determining the quality of the required approximation. The problems considered include the approximate maximum problem, approximate sorting and approximate merging. Our results imply as special cases, all the known results about the time complexity for parallel sorting, parallel merging and parallel selection of the maximum (in the comparison model), up to a constant factor. We mention one very special but representative result concerning the approximate maximum problem; suppose we wish to find, among the given n elements, one which belongs to the biggest $n/2$, where in each round we are allowed to ask n binary comparisons. We show that $\log^* n + O(1)$ rounds are both necessary and sufficient in the best algorithm for this problem.

1. Introduction

1.1 The model and previous results

Parallel comparison algorithms received a lot of attention during the last decade. The problems considered include sorting ([1], [2], [5], [6], [9], [10], [13], [17], [20], [22], [24], [25], [26], [27], [30]), merging ([20], [23], [25], [29], [30]), selecting ([1], [7], [12], [27], [30]) and approximate sorting ([1], [5], [8], [14], [16]). The common model of computation considered in the parallel comparison model, introduced by Valiant [30], where only comparisons are counted. In this model, during each time unit (called a *round*) a set of binary comparisons is performed. The actual set of comparisons asked is chosen according to the results of the comparisons done in the previous rounds. The objective is to solve the problem at hand, trying to minimize the number of comparison rounds as well as the number of comparisons performed in each round. Note that this model ignores the time corresponding to deducing consequences from comparisons performed, as well as communication and memory addressing time. However, in some situations, the comparisons cost more than the rest of the algorithm and hence this seems to be the relevant model. Moreover, any lower bound here, applies to any comparison based algorithm.

Let n denote the number of elements we have (from a totally ordered domain), and suppose we have p parallel processors, i.e., we are allowed to perform p comparisons in each round. The (worst-case) time complexity of the best deterministic

AMS subject classification (1980): 68 E 05

*Research supported in part by Allon Fellowship, by a Bat Sheva de Rothschild grant and by the Fund for Basic Research administered by the Israel Academy of Sciences.

algorithm for each of the basic comparison problems is known, up to a constant factor, for all admissible values of n and p. For sorting, this time is $\Theta(\log n/\log(1+p/n))$, as shown in [9], [13], [5] (and as proved in [2] the same bounds hold for the average-case complexity, as well). Here, and throughout the paper, the notation $g(n) = \Theta(f(n))$ means, as usual, that $g(n) = O(f(n))$ and $f(n) = O(g(n))$. For finding the maximum the time complexity is $\Theta(n/p + \log((\log n)/(\log(2+p/n)))$, as shown in [30], and the results of [7] and [12] show that the same bounds hold for general selection. Finally, the time complexity for merging two sorted lists of n elements each is $\Theta(n/p + \log((\log n)/(\log(2+p/n)))$, as proved in [30], [20].

All the above problems are special cases of the more general corresponding approximation problems. In these problems, one is satisfied with an approximate solution of the problem at hand. Thus, for example, in approximate sorting we wish to know all the order relations between pairs of elements but at most εn^2, where $1/(2n^2) \leq \varepsilon \leq 1/4$ is a given accuracy parameter (which may depend on n). The problem of approximate merging and that of finding an approximate maximum are defined similarly, as described in detail in the next subsection. Notice that approximate sorting with $\varepsilon = 1/(2n^2)$ corresponds to usual sorting, and hence a solution to the approximation problem contains the result for sorting as a special case. Similarly, each of the other approximation problems is more general than the corresponding problem it approximates.

There are several known results about approximate sorting, most of which deal with the minimum number of comparisons $p = p(n)$ that suffices to determine all order relations between pairs but at most $o(n^2)$ in one round. See [16], [1], [8], [14]. The problem of finding an approximate maximum also arises naturally in various situations, and as we show below can be solved considerably more efficiently than that of finding the exact maximum.

1.2 The main results

We determine, up to a constant factor, the time complexity for finding an approximate maximum, for approximate sorting and for approximate merging for all admissible values of the three parameters n, p and ε. This implies, as a special case, all the known results about the time complexity of the corresponding comparison problems. In addition, it reveals certain surprising differences between the time complexities of some of the problems and these of their approximation generalizations. All our upper bounds are obtained by explicit algorithms that apply several known explicit expanders, and only the constants can be somewhat improved by using random graphs instead of explicit ones. We next state our results in this full generality. The functions appearing in these results are somewhat complicated, and hence it is not easy to see the exact implications of the theorems below. It is thus worth mentioning, before the statements of the theorems below, one somewhat surprising special case which appears in [3] and answers a question raised by N. Pippenger and by J. Komlós. Suppose we wish to find, among our n elements, an element which belongs to the biggest $n/2$, where in each round we allow n comparisons. We show in [3] that $\log^* n - 4$ rounds are necessary and $\log^* n + 2$ are sufficient for this problem. Here $\log^* n$ denotes the minimum number k such that, starting with n, k applications of logarithms in base 2 suffice to reach a number smaller than or equal to 1.

We now turn to the general problems.

For integers $n \geq 2$ and p, $1 \leq p \leq \binom{n}{2}$, and for a real number ε, $1/n \leq \varepsilon \leq 1/2$, let $r(n,p,\varepsilon)$ denote the time complexity of the best deterministic comparison algorithm that finds, among n elements, an element whose rank belongs to the top εn ranks, using p comparisons in each round. The case $\varepsilon = 1/2$ corresponds to finding an element in the top $n/2$ ranks and hence the result stated above and proved in [3] is:

Proposition 1.0. $\log^* n - 4 \leq r(n,n,1/2) \leq \log^* n + 2$.

For $\varepsilon = 1/n$, the problem is that of finding the exact maximum, and the case $p = 1$ corresponds to serial algorithms. The general case is the following:

Theorem 1.1. *For all admissible n, p, ε*

$$r(n,p,\varepsilon) = \Theta\left(\frac{n}{p} + \log\frac{\log(1/\varepsilon)}{\log(2+p/n)} + \log^* n - \log^*\left(1+\frac{n}{p}\right)\right).$$

Thus for all n, $p \leq 2n$, ε

$$r(n,p,\varepsilon) = \Theta\left(\frac{n}{p} + \log\log\frac{1}{\varepsilon} + \log^* n\right)$$

and for all n, $p \geq 2n$, ε

$$r(n,p,\varepsilon) = \Theta\left(\log\frac{\log(1/\varepsilon)}{\log(p/n)} + \log^* n - \log^*(p/n)\right).$$

For $\varepsilon = 1/n$ this theorem reduces to Valiant's result about finding the maximum [30]. For $\varepsilon = 1/2$, $p = n$ this reduces to our Proposition 1.0 (with a somewhat cruder estimate).

Next we consider approximate sorting. For $n \geq 2$, $1 \leq p \leq \binom{n}{2}$, and $1/(2n^2) \leq \varepsilon \leq 1/4$, let $a(n,p,\varepsilon)$ denote the time complexity of the best deterministic comparison algorithm, that uses p comparisons in each round and finds, given n elements, all the order relations between pairs but at most εn^2. The results of [16], [1], [8], [14] deal with the minimum p for which $a(n,p,\varepsilon) = 1$ for some $\varepsilon = o(1)$. Note that a precise determination of $a(n,p,\varepsilon)$ contains all the known results about the time complexity of deterministic comparison sorting or approximate sorting algorithms. The following result determines $a(n,p,\varepsilon)$ up to a constant factor, for all possible n, p, ε.

Theorem 1.2. *For all admissible n, p, ε*

$$a(n,p,\varepsilon) = \Theta\left(\frac{\log(1/\varepsilon)}{\log(1+p/n)} + \log^* n - \log^*\left(1+\frac{p}{n}\right)\right).$$

Thus, for $p \leq 2n$,

$$a(n,p,\varepsilon) = \Theta\left(\frac{n\log(1/\varepsilon)}{p} + \log^* n\right),$$

and for $p \geq 2n$,

$$a(n,p,\varepsilon) = \Theta\left(\frac{\log(1/\varepsilon)}{\log(p/n)} + \log^* n - \log^*\left(\frac{p}{n}\right)\right).$$

For $\varepsilon = 1/(2n^2)$ this theorem corresponds to sorting, and gives the known

$$\Theta(\log n/\log(1+p/n))$$

bound (which is $\Theta((n\log n)/p)$ for $p \leq 2n$ and is $\Theta(\log n/\log(p/n))$ for $p \geq 2n$), (see [9], [13], [5]). Notice that for $p = n$ and for any $\varepsilon \geq 1/2^{\log^* n}$ $a(n,n,\varepsilon) = \Theta(\log^* n)$. By Theorem 1.1, $\Omega(\log^* n)$ rounds are required (with $p = n$) even if we wish to find one element known to be greater than $n/2$ others. By the last equality, $O(\log^* n)$ rounds are already sufficient to get almost all the order relations between pairs.

Finally, we consider the problem of approximate merging. In this case the results and the methods are simpler, (the function $\log^*$ does not appear in the statement of the result), and are similar to the methods of [30], [20]. For n, $1 \leq p \leq n^2$ and $1/n^2 \leq \varepsilon \leq 1/2$, let $m(n,p,\varepsilon)$ denote the time complexity of the best comparison merging algorithm, that uses p comparisons in each round and finds, given 2 sorted lists, each of size n, all the order relation between pairs but at most εn^2.

The results of [30], [20] deal with full merging, i.e. the case $\varepsilon < 1/n^2$. The following theorem determines $m(n,p,\varepsilon)$, up to a constant factor, for all admissible n, p, ε.

Theorem 1.3. *For all admissible* n, p *and* $1/n \leq \varepsilon \leq 1/2$

$$m(n,p,\varepsilon) = \Theta\left(\frac{1}{\varepsilon p} + \log\frac{\log 1/\varepsilon}{\log(2+\varepsilon p)}\right).$$

Thus for $p \leq 2/\varepsilon$, $m(n,p,\varepsilon) = \Theta(1/(\varepsilon p) + \log\log 1/\varepsilon)$ and for $p \geq 2/\varepsilon$,

$$m(n,p,\varepsilon) = \Theta\left(\log\frac{\log 1/\varepsilon}{\log \varepsilon p}\right).$$

For the case $\varepsilon \leq 1/n$, the bounds are the same as for $\varepsilon = 1/n$ (up to a constant factor), which are the same bounds as for exact merging:

$$\Theta\left(\frac{n}{p} + \log\frac{\log n}{\log(2+p/n)}\right).$$

1.3 Consequences of the results

As already mentioned, Theorems 1.1, 1.2 and 1.3 include, as special cases, all the known results for the time complexities of deterministic parallel comparison algorithms for sorting, merging and finding the maximum, up to a constant factor. However, it seems that the most interesting consequence of these theorems is the fact that some of the approximation problems can be solved much more efficiently than their precise versions. This corresponds to the $\log^*$ terms that appear in the results

for the approximation problems. To be specific, consider, for example, the special case considered in Proposition 1.0. This corresponds to the *approximate maximum problem*, i.e., the problem of finding, among n elements, an element whose rank belongs to the top $n/2$ ranks, using n comparisons in each round. It is trivial to show that in the serial comparison model this problem requires $n/2$ comparisons: only a constant factor better than the problem of finding the exact maximum. It is therefore rather surprising that with n comparisons in each round this problem can be solved much faster than that of finding the exact maximum in the same conditions. As shown in Proposition 1.0, $\log^* n + \Theta(1)$ rounds are both necessary and sufficient for finding an approximate maximum among n elements, using n comparisons in each round. This is considerably faster than the best algorithm for finding the exact maximum with n comparisons in each round, which requires, as shown in [30], $\log\log n + \Theta(1)$ rounds. Moreover, as shown in Theorem 1.1, $O(\log^* n)$ rounds suffice to find an element in the top $n/2^{2^{\log^* n}}$ ranks, i.e., a rather good approximation for the maximum (and, in fact, by Theorem 1.2 that many rounds suffice for finding good approximation for any other rank). In several cases, the parallel comparison model seems to be the relevant model. An example is the test of consumer preferences among n items (see [28]). If we wish to find the best choice of a consumer (with n comparisons in each round) $\log\log n + \Theta(1)$ rounds are required. On the other hand, if we are satisfied with the more modest choice of an almost best candidate (say, finding an item in the top $n/1,000,000$ ones), $\log^* n + \Theta(1)$ rounds suffice (and are also necessary). As our algorithm for the upper bound can be described explicitly, such a choice can actually be done in such a small number of rounds.

We say that a parallel algorithm achieves *optimal speed up* if the product of its running time by the number of processors it uses is equal, up to a constant factor, to the running time of the best serial algorithm for the same problem. I.e., if $T(n) \cdot p(n) = O(\mathrm{Seq}(n))$, where $p(n)$ is the number of processors, $T(n)$ and $\mathrm{Seq}(n)$ are the running times of the parallel algorithm and the best serial one, respectively, and n is the size of the input. It is easy to see that if $T'(n) > T(n)$ and there is an optimal speed up algorithm with running time $T(n)$, then there is also an optimal speed up algorithm for the same problem with running time $T'(n)$. The *parallelism break point* of a problem is the minimum $T(n)$ so that there is an optimal speed up algorithm with running time $T(n)$. A considerable amount of effort in the study of parallel algorithms is done in attempts of trying to identify the break points of various algorithmic problems. The break point for sorting n elements (in the comparison model) is $\Theta(\log n)$, as follows from the results of [9], [5], [13]. The break point of merging two lists of size n is $\Theta(\log\log n)$, (see [20], [25]), and the break point for selection is also $\Theta(\log\log n)$, (see [30], [7], [12]). Theorems 1.1, 1.2 and 1.3 supply the break points of each of the approximation problems considered here. Notice that as the accuracy parameter ε varies so does the corresponding problem and its break point. Consequently, we obtain the previously known break points (and, in particular, for the extreme values of ε, we obtain the previously known break points for the non-approximation problems, mentioned above). As a special case let us note that Theorem 1.1 shows that $\Theta(\log^* n)$ is the parallelism break point of the approximate maximum problem, i.e., of the problem of finding an element among the top $n/2$ ones.

The rest of this paper is organized as follows: Section 2 includes the proofs of the lower bounds in all the theorems. In Section 3 the corresponding upper bounds are proved. Section 4 contains some concluding remarks and results about approximate selection, where the exact complexity is still open. The proofs of Sections 2 and 3 are quite lengthy and complicated. They combine certain probabilistic arguments and results from Extremal Graph Theory, with various properties of random graphs (or explicit expanders) and several known results about selecting and sorting in rounds.

2. The Lower Bounds

In this section we prove the lower bounds for all the problems, i.e., for finding the approximate maximum, for approximate sorting and at the end for approximate merging. We split the proofs into several theorems and lemmas.

We start (2.1–2.5) with a crucial special case for the approximate maximum problem; $p \geq n$ and $\varepsilon = 1/2$. Define $\alpha = p/n$. We show that in this case $\log^* n - \log^* \alpha - O(1)$ rounds are needed. The proof here is a modified version of the one given in our previous paper [3], which considers the case $p = n$ and $\varepsilon = 1/2$. Afterwards we complete the proof of the lower bound by combining the proof for this case with a modification of Valiant's lower bound (2.6) and the serial lower bound for the maximum problem. Next we consider approximate sorting, prove a serial lower bound (2.7), a lower bound that deals with algorithms that end after k rounds (2.8) and complete the proof by combining these bounds (2.9) with the approximate maximum bounds. Finally we deal with approximate merging. We prove a serial lower bound (2.10) and a lower bound for $p \geq 4/\varepsilon$ (2.11) and combine them to get the desired lower bound.

The case $p = n$ and $\varepsilon = 1/2$ of the approximate maximum problem is considered in [3]. The proof of the lower bound for the case $p \geq n$ is very similar, but contains several additional complications and is presented below. As usual we define, for $a \geq 1$ and $k \geq 0$, $a^{(k)}$ by $a^{(0)} = 1$ and $a^{(k)} = a^{a^{(k-1)}}$ for $k \geq 1$ and put $\log^* n = \min\{k : 2^{(k)} \geq n\}$. We also define for α, $a \geq 1$ and $k \geq 1$ $a^{(k,\alpha)}$ by $a^{(1,\alpha)} = \alpha a$ and $a^{(k,\alpha)} = a^{a^{(k-1,\alpha)}}$ for $k \geq 2$.

There is an obvious, useful correspondence that associates each round of any comparison algorithm in the parallel comparison model with a graph whose set of vertices is the set of elements we have. The (undirected) edges of this graph are just the pairs compared during the round. The answer to each comparison corresponds to orienting the corresponding edge from the larger element to the smaller. Thus in each round we get an acyclic orientation of the corresponding graph, and the transitive closure of the union of the r oriented graphs obtained until round r represents the set of all pairs of elements whose relative order is known at the end of round r.

It is convenient to establish the lower bound by considering the following (full information) game, called the *orientation game*, and played by two players, the *graphs player* and the *order player*. Let V be a fixed set of n vertices. The game consists of *rounds*. In the first round the graphs player presents an undirected graph G_1 on V with at most αn edges and the order player chooses an acyclic orientation H_1 of G_1, and shows it to the graphs player, thus ending the first round. In the second round the graphs player chooses again, an undirected graph G_2 with at most

αn edges on V, and the order player gives it an acyclic orientation H_2, consistent with H_1 (i.e., the union of H_1 and H_2 is also acyclic), which he presents to the graphs player. The game continues in the same manner; in round i the graphs player chooses an undirected graph G_i with at most αn edges on V, and the order player gives it an acyclic orientation H_i, such that the union $H_1 \cup \ldots \cup H_i$ is also acyclic. The game ends when, after, say, round r, there is a vertex v in V whose outdegree in the *transitive closure* of $H_1 \cup \ldots \cup H_r$ is at least $n/2$. The objective of the graphs player is to end the game as early as possible, and that of the order player is to end it as late as possible. The following fact states the (obvious) connection between the orientation game and approximate maximum problem.

Proposition 2.1. *The graphs player can end the orientation game in r rounds if and only if there is a comparison algorithm that finds an approximate maximum among n elements (i.e., an element whose rank is in the top $n/2$ ranks), using αn comparisons in each round, in r rounds.* ∎

In view of the last proposition, a proof of existence of a strategy for the order player that enables him to avoid ending the orientation game in r rounds implies that $r+1$ is a lower bound for the time complexity of the approximate maximum problem.

The next proposition is our main tool for establishing the existence of such a strategy for $r = \log^* n - \log^* \alpha - 5$.

Proposition 2.2. *There exists a strategy for the order player to maintain, for every $d \geq 1$, the following property $P(d)$ of the directed acyclic graph constructed during the game.*

Property $P(d)$: Let $H(d) = H_1 \cup \ldots \cup H_d$ be the union of the oriented graphs constructed in the first d rounds. Then there is a subset $V_0 \subseteq V$ of size at most

$$|V_0| \leq \frac{n}{8} + \frac{n}{16} + \ldots + \frac{n}{2^{d+2}}$$

and a proper $D = 2048^{(d,\alpha)}$-vertex-coloring of the induced subgraph of $H(d)$ on $V - V_0$ with color classes $V_1, V_2, \ldots, V_D$ (some of which may be empty), such that for each $i > j \geq 1$ and each $v \in V_i$, v has at most 2^{i-j-2} neighbors in V_j. Furthermore; for every $i > j \geq 0$ any edge of $H(d)$ that joins a member of V_i to a member of V_j is directed from V_i to V_j.

Proof. We apply induction on d. For $d = 1$, the graph $G_1 = (V, E_1)$ constructed by the graphs player has at most $n\alpha$ edges. Let V_{00} be the set of all vertices in V whose degree is at least 32α. Clearly

$$|V_{00}| \leq n/16 \tag{2.1}$$

Put $U = V - V_{00}$ and let K be the induced subgraph of G_1 on U. As the maximum degree in K is less than 32α, K has, by a standard, easy result from extremal graph theory (see, e.g., [15, pp.222]) a proper vertex — coloring by 32α colors and hence, certainly, a proper vertex coloring by 2048α colors. Let $U_1, U_2, \ldots, U_{2048\alpha}$ be the color classes. For every vertex u of K, let $N(u)$ denote the set of all its neighbors in K. For a permutation π of $1, 2, \ldots, 2048\alpha$ and any vertex u of K define the π-degree

$d(\pi, u)$ of u as follows: let i satisfy $u \in U_{\pi(i)}$ then $d(\pi, u) = \sum_{j=1}^{i-1} |N(u) \cap U_{\pi(j)}|/2^{i-j}$. We claim that the expected value of $d(\pi, u)$ over all permutations π of $\{1, \ldots, 2048\alpha\}$, is at most $32/2048 = 1/64$. Indeed, for a random permutation π the probability that a fixed neighbor v of u contributes $1/2^r$ to $d(\pi, u)$ is at most $1/(2048\alpha)$ for every fixed $r > 0$. Hence, each neighbor contributes to this expected value at most $1/(2048\alpha) \sum_{r>0} 1/2^r = 1/(2048\alpha)$ and the desired result follows, since $|N(u)| < 32\alpha$.

Consider now the sum $\sum_{u \in U} d(\pi, u)$. The expected value of this sum (over all π's) is at most $|U|/64$, by the preceding paragraph. Hence, there is a fixed permutation σ such that $\sum_{u \in U} d(\sigma, u) \leq |U|/64$. Put $V_{01} = \{u \in U \mid d(\sigma, u) > 1/4\}$. Clearly

$$|V_{01}| \leq 4 \cdot |U|/64 \leq |U|/16 \leq n/16.$$

Define $V_0 = V_{00} \cup V_{01}$, $W = U - V_{01} = V - V_0$. The last inequality together with inequality (2.1) gives

$$|V_0| \leq n/8 .$$

Let F be the induced subgraph of G_1 on W and define $V_i = U_{\sigma(i)} \cap W$ $(1 \leq i \leq 2048\alpha)$. The V_i's clearly form a proper vertex coloring of F. Also, for every i, $1 \leq i \leq 2048\alpha$ and every $v \in V_i$

$$\sum_{j=1}^{i-1} |N(v) \cap V_j|/2^{i-j} < 1/4$$

and hence v has at most 2^{i-j-2} neighbors in V_j for each j, $1 \leq j < i$. Let H_1 be any acyclic orientation of G_1 in which all edges that join a member of V_i to a member of V_j, where $i > j \geq 1$, are directed from V_i to V_j (the edges inside V_0 can be directed in an arbitrary acyclic manner). Clearly $H(1) = H_1$ satisfies the property $P(1)$. Thus, the order player can orient G_1 according to H_1. This completes the proof of the case $d = 1$.

Continuing the induction, we now assume that $H(r)$ has property $P(r)$ for all $r < d$, and prove that the order player can always guarantee that $H(d)$ will have property $P(d)$. We start by proving the following simple lemma.

Lemma 2.3. *Let F be a directed acyclic graph with a proper g-vertex coloring with color classes $W_1, W_2, \ldots, W_g$. Suppose that for each $g \geq i > j \geq 1$ and each $v \in W_i$, v has at most 2^{i-j-2} neighbors in W_j, and that every edge of F whose ends are in W_i and W_j for some $i > j$ is directed from W_i to W_j. Then the outdegree of every vertex of F in the transitive closure of F is smaller than 4^g.*

Proof. Let v be an arbitrary vertex of F. The outdegree of v in the transitive closure of F is obviously smaller than or equal to the total number of directed paths in F that start from v. Suppose $v \in W_i$. Each such directed path must be of the from v, $v_{i_2}, v_{i_3}, \ldots, v_{i_r}$, where $i > i_2 > i_3 > \ldots > i_r \geq 1$, $v_{i_2} \in W_{i_2}, \ldots, v_{i_r} \in W_{i_r}$. There are 2^{i-1} possibilities for choosing $i_2, i_3, \ldots, i_r$. Also, as each vertex of the path is a neighbor of the previous one, there are at most $^{i-i_2-2}$ possible choices for v_{i_2},

$2^{i_2-i_3-2}$ possible choice for v_{i_3} (for each fixed choice of v_{i_2}), etc. Hence, the total number of paths is at most $2^{i-1} \cdot 2^{i-i_2-2} \cdot 2^{i_2-i_3-2} \cdot \ldots \cdot 2^{i_r-1-i_r-2} < 2^g \cdot 2^{i-i_r} < 4^g$. This completes the proof of the lemma. ∎

Returning to the proof of Proposition 2.2, recall that $d \geq 2$ and that by the induction hypothesis $H(d-1)$ has property $P(d-1)$. Thus, there is a subset $V_0 \subseteq V$ satisfying

$$|V_0| \leq \frac{n}{8} + \frac{n}{16} + \cdots + \frac{n}{2^{d+1}} \tag{2.2}$$

and a proper $D = 2048^{(d-1,\alpha)}$-vertex-coloring of the induced subgraph of $H(d-1)$ on $V - V_0$ with color classes $V_1, V_2, \ldots, V_D$ satisfying the conditions of property $P(d-1)$. Put $U = V - V_0$, let F be the induced subgraph of $H(d-1)$ on U and let $T = (U, E(T))$ be the transitive closure of F. Let $G_d = (V, E_d)$ be the graph constructed by the graphs player in round number d. Let $\overline{V}_{00}$ be the set of all vertices in U whose degree in G_d is at least $\alpha \cdot 2^{d+4} \cdot 4^D$ and define

$$V_{00} = \overline{V}_{00} \cup \{u \in U \,:\, \exists v \in \overline{V}_{00} \text{ with } (v,u) \in E(T)\} .$$

Since G_d has at most $n\alpha$ edges, $|\overline{V}_{00}| \leq n\alpha/(\alpha 2^{d+3} \cdot 4^D)$. Also, by Lemma 2.3, the outdegree of each $v \in \overline{V}_{00}$ in T is at most $4^D - 1$. Hence

$$|V_{00}| \leq n/2^{d+3} . \tag{2.3}$$

Let $\overline{G}$ be the induced subgraph of G_d on $U - V_{00}$. Then the maximum degree in $\overline{G}$ is smaller than $\alpha \cdot 2^{d+4} \cdot 4^D$. For each i, $1 \leq i \leq D$, let $\overline{G}^i$ denote the induced subgraph of $\overline{G}$ on $(U - V_{00}) \cap V_i$. As each $\overline{G}^i$ is a subgraph of $\overline{G}$, it has a proper vertex coloring with $\alpha 2^{d+4} \cdot 4^D$ colors. For each i, $1 \leq i \leq D$, fix a proper n_i-vertex-coloring of $\overline{G}^i$ with color classes $U_{N_i+1}, U_{N_i+2}, \ldots, U_{N_i+n_i}$ (some of which may be empty) where $N_i = \sum_{j=1}^{i-1} n_j$ and

$$n_i \geq 100 \cdot 2^{2d+7} \cdot 16^D \alpha \text{ for each } 1 \leq i \leq D \text{ and } \sum_{i=1}^{D} n_i = 2048^D . \tag{2.4}$$

(Notice that since $D = 2048^{(d-1,\alpha)}$, $d \geq 2$, $\alpha \leq D \leq 2^D$ there is such a choice for the n_i's). For every vertex u of $\overline{G}$, let $N(u)$ denote the set of all its neighbors in $\overline{G}$. Let us call a permutation π of 1, 2, 3, ..., $\sum_{i=1}^{D} n_i$ *legal* if it maps each set of the form $\{N_i + 1, \ldots, N_i + n_i\}$ into itself (and only permutes the elements inside these sets among themselves). For any vertex u of $\overline{G}$ and any legal permutation π, define the π – degree $d(\pi, u)$ as follows; let k satisfy $u \in U_{\pi(k)}$, then

$$d(\pi, u) = \sum_{j=1}^{k-1} |N(u) \cap U_{\pi(j)}|/2^{k-j} .$$

We claim that the expected value of $d(\pi, u)$ over all $\prod_{i=1}^{D} n_i!$ legal permutations, is at most $|N(u)|/\min_{1\le i\le D} n_i \le 1/(100\cdot 2^{d+3}\cdot 4^D)$. Indeed, consider a fixed neighbor v of u. If v belongs, like u, to the same graph $\overline{G}^k$, then the probability that for a random legal permutation π, v will contribute $1/2^r$ to $d(\pi, u)$ is at most $1/n_k$, for each fixed $r > 0$. Otherwise, it is easy to check that this probability is even smaller. Hence, each neighbor contributes to this expected value at most $\left(\sum_{r>0} 1/2^r\right)/n_k = 1/n_k$, and the claim follows.

Consider now the sum $\sum d(\pi, u)$, where u ranges over all vertices of $\overline{G}$. The expected value of this sum (over all permutations π) is at most $|V(\overline{G}|/(100 \cdot 2^{d+3} \cdot 4^D) \le n/(100 \cdot 2^{d+3} \cdot 4^D)$. Hence, there is a fixed legal permutation σ such that $\sum_{u\in V(\overline{G})} d(\sigma, u) \le n/(100 \cdot 2^{d+3} \cdot 4^D)$. Define $\overline{V}_{01} = \{u \in V(\overline{G}) : d(\sigma, u) > 1/100\}$ and $V_{01} = \overline{V}_{01} \cup \{u \in V(\overline{G}) : \exists v \in \overline{V}_{01} \text{ with } (v, u) \in E(T)\}$. Clearly $|\overline{V}_{01}| \le n/(2^{d+3} \cdot 4^D)$ and hence, by Lemma 2.3,

$$|V_{01}| \le n/2^{d+3}. \tag{2.5}$$

Put $V_0' = V_0 \cup V_{00} \cup V_{01}$, $W = V - V_0$. By (2.2), (2.3) and (2.5)

$$|V_0'| \le \frac{n}{8} + \frac{n}{16} + \ldots + \frac{n}{2^{d+2}}.$$

Let $\tilde{G}$ be the induced subgraph of $\overline{G}$ on W and define $V_i' = U_{\sigma(i)} \cap W$ $(1 \le i \le 2048^D = 2048^{(d,\alpha)})$. The sets V_i' clearly form a proper vertex coloring of $\tilde{G}$. Moreover, as each U_k is an independent set in $H(d-1)$, the sets V_i' actually form a proper vertex coloring of $H(d-1)$, as well. Moreover, for every i, $1 \le i \le 2048^{(d,\alpha)}$ every $v \in V_i'$ satisfies

$$\sum_{j=1}^{i-1} |N(v) \cap V_j'|/2^{i-j} \le 1/100,$$

where $N(v)$ is the set of all neighbors of v in $\overline{G}$. Thus, for each fixed j, $1 \le j < i$, v has at most $2^{i-j}/100$ neighbors in V_j'. Let H_d be any acyclic orientation of the edges of G_d obtained by orienting all the edges that join a member of V_i' and a member of V_j', where $i > j \ge 0$, from V_i' to V_j'. The edges inside V_0' are oriented in an arbitrary acyclic order consistent with the order given on $H(d-1)$. Notice that all the edges of $H(d-1)$ that do not lie inside V_0' are also oriented from V_i' to V_j' with $i > j \ge 0$. In order to show that $H(d) = H(d-1) \cup H_d$ has the property $P(d)$, it remains to check that for every $i > j \ge 1$; every $v \in V_i'$ has at most 2^{i-j-2} neighbors in V_j'. By the construction, v has at most $2^{i-j}/100$ neighbors in V_j' in the new graph H_d. Recall that each V_i' is a subset of one of the sets V_k corresponding to the graph $H(d-1)$. Suppose $V_i' \subseteq V_k$, $V_j' \subseteq V_l$. Then $k \ge l$. If $l = k$ or $l = k-1$ then, since v has at most $\lfloor 2^{k-l-2} \rfloor = 0$ neighbors in V_l in the graph $H(d-1)$, it follows that in $H(d)$ v

has at most $2^{i-j}/100 \leq 2^{i-j-2}$ neighbors in V'_j, as needed. If $l \leq k-2$, observe that our construction implies that

$$(i-j) \geq (k-l-1) \min_{1 \leq i \leq D} n_i \geq (k-l-1) \cdot 100 \cdot 2^{2d+7} \cdot 16^D > (k-l) \cdot 100 \geq 200.$$

Thus, in this case the total number of neighbors of v in V'_j is at most $2^{i-j}/100 + 2^{k-l-2} \leq 2^{i-j}/100 + 2^{(i-j)/100} \leq 2^{i-j-2}$.

We conclude that the order player can orient G_d according to H_d, and maintain the property $P(d)$ of the graph $H(d) = H(d-1) \cup H_d$. This completes the induction and the proof of Proposition 2.2. ∎

The result stated in Theorem 2.5 below, is an easy consequence of Proposition 2.2. and the following simple lemma.

Theorem 2.4. *For every* $d \geq 1$, $2^{(d+3+\log^* \alpha)} \geq 32 \cdot 2048^{(d,\alpha)}$.

Proof. We apply induction on d. For $d=1$ the inequality is trivial as $2^{(4+\log^* \alpha)} \geq 2^{(4)\cdot(\log^* \alpha)} \geq 2^{16} \cdot \alpha = 32 \cdot 2048^{(1,\alpha)}$. Assuming it holds for d–1, we prove it for $d \geq 2$. By assumption $2^{(d+2+\log^* \alpha)} \geq 32 \cdot 2048^{(d-1,\alpha)}$. Hence $2^{(d+3+\log^* \alpha)} = 2^{2^{(d+2+\log^* \alpha)}} \geq 2^{32 \cdot 2048^{(d-1,\alpha)}} = (2^{21} \cdot 2^{11})^{2048^{(d-1,\alpha)}} = (2^{21})^{2048^{(d-1,\alpha)}} \cdot (2048)^{2048^{(d-1,\alpha)}} > 32 \cdot (2048)^{(d,\alpha)}$. ∎

Theorem 2.5. *The order player can avoid ending the orientation game during the first* $\log^* n - \log^* \alpha - 5$ *rounds. Hence, by Proposition 2.1, the time required for finding an approximate maximum among* n *elements using* αn *comparisons in each round is at least* $\log^* n - \log^* \alpha - 4$.

Proof. Clearly we may assume that $\log^* n - \log^* \alpha - 5 \geq 0$. By Proposition 2.2, the order player can maintain the property $P(d)$ for each of the graphs $H(d)$ constructed during the algorithm. Notice that by Lemma 2.3, the outdegree of every vertex in the transitive closure of a graph that satisfies $P(d)$ is at most $4^D + n/8 + n/16 + \ldots + n/(2^{d+2}) < 4^D + n/4$, where $D = 2048^{(d,\alpha)}$. It thus follows that if $4^{2048^{(r,\alpha)}} \leq n/4$ then the graphs player can keep playing for at least $r+1$ rounds. Therefore, by Lemma 2.4, the assertion of the theorem will follow if for $r = \log^* n - \log^* \alpha - 5$ the inequality $4^{2^{(r+3+\log^* \alpha)}}/32 \leq n/4$ holds. Since for $r > 0$ $4 \cdot 4^{2^{(r+3+\log^* \alpha)}}/32 < 2^{(r+4+\log^* \alpha)}$ this follows immediately from the definition of $\log^* n$. ∎

Lemma 2.6. *For* $p \geq 2n$, $r(n,p,\varepsilon) = \Omega\left(\log \frac{\log 1/\varepsilon}{\log p/n}\right)$.

Proof. The proof is an easy modification of Valiant's proof for the maximum problem (see [30]).

If the algorithm consists of s rounds and m denotes the number of candidates for the maximum after these s rounds, the adversary can ensure that $m/(m+2p) \geq$

$(n/(n+2p))^{2^s}$. (This follows easily from Turán's Theorem, as shown in [30]). But clearly $m \leq \varepsilon n$, therefore, since $p \geq 2n$

$$s \geq \log \frac{\log \frac{m+2p}{m}}{\log \frac{n+2p}{n}} = \Omega\left(\log \frac{\log p/m}{\log p/n}\right) = \Omega\left(\log \frac{\log(p/n) + \log(1/\varepsilon)}{\log(p/n)}\right)$$
$$\geq \Omega\left(\log \frac{\log 1/\varepsilon}{\log p/n}\right) \qquad \blacksquare$$

Proof of the lower bound of Theorem 1.1.

Clearly at least $(1-\varepsilon)n \geq n/2$ comparisons are needed, even in the serial case, to conclude that an element belongs to the top εn ones. Hence $r(n,p,\varepsilon) \geq n/(2p) = \Omega(n/p)$, for every $p \geq 1$. A lower bound of $\Omega\left(\log \frac{\log 1/\varepsilon}{\log p/n}\right)$ for $p \geq 2n$ follows from Lemma 2.6 and the bound $\Omega(\log\log 1/\varepsilon)$ for $p \leq 2n$ is the lower bound from that lemma even for $p = 2n$.

The $\Omega(\log^* n - \log^*(1+p/n))$ term follows from Theorem 2.5 for $p \geq n$ (even for $\varepsilon = 1/2$). For $p \leq n$ we simply take the bound of Theorem 2.5 for $p = n$ and $\varepsilon = 1/2$. $\blacksquare$

Theorem 2.7. *Any serial algorithm that finds all but at most εn^2 of the order relations between n elements ($1/n^2 \leq \varepsilon \leq 1/4$) needs at least $\Omega(n\log(1/\varepsilon))$ rounds.*

Proof. The proof is by a simple counting argument. For, say, $\varepsilon > 1/100$ the assertion is trivial (since at least one element is known to be in the top $0.8n$ ones). We thus assume $\varepsilon \leq 1/100$. First, we estimate the number of orders that fit one given output of the algorithm.

If we have all the order relations but εn^2 of them, then there are at least $n/2$ elements whose relative order to all but $2\varepsilon n$ elements is known. Hence, the number of orders consistent with these relations is at most

$$\binom{n}{\frac{n}{2}} \left(\frac{n}{2}\right)! \cdot (2\varepsilon n)^{n/2} = \frac{n!(2\varepsilon n)^{n/2}}{\left(\frac{n}{2}\right)!}.$$

Therefore the number of distinct outputs of the algorithm is at least

$$\frac{n!}{\frac{n!}{\left(\frac{n}{2}\right)!}(2\varepsilon n)^{n/2}} = \frac{\left(\frac{n}{2}\right)!}{(2\varepsilon n)^{n/2}} \geq \frac{\left(\frac{n}{2e}\right)^{n/2}}{(2\varepsilon n)^{n/2}} = \left(\frac{1}{4e\varepsilon}\right)^{n/2}.$$

Hence the number of rounds needed is at least

$$\log\left(\frac{1}{4e\varepsilon}\right)^{n/2} = \frac{n}{2}\log\left(\frac{1}{4e\varepsilon}\right) = \Omega(n\log(1/\varepsilon)) \qquad \blacksquare$$

Define $c(k,n,m)$ to be the total number of comparisons needed to sort n elements in k rounds up to at most m unknown order relations between pairs, $0 \leq m \leq \binom{n}{2}$.

Theorem 2.8. *For all possible* n, m *and* $k \geq 1$, $c(k,n,m) > k\left(\dfrac{n^{1+1/k}}{d(1+m)^{1/(2k)}} - n\right)$ *where* $d = 16\sqrt{2}$.

Proof. By induction. We leave the base of the induction to the end.

The inductive assumption: Given k, n, if $k' = k$ and $n' < n$, or $k' < k$ and $n' \leq n$ then for every m'

$$c(k',n',m') > k'\left(\frac{n'^{1+1/k'}}{d(1+m')^{1/(2k')}} - n'\right)$$

Take any k-round algorithm for sorting a set V of n elements. The first round of the algorithm consists of some set E of comparisons. As usual look at them as edges in the graph $G = (V,E)$. An independent set is *maximal* if it is not a proper subset of another independent set. Consider the graph of the first round of comparisons. Let S be a maximal independent set in this graph and denote $x = |S|$. Each of the $n - x$ elements of $\overline{S}$ must share an edge with an element of S, otherwise S is not maximal. For our lower bound, we restrict our attention to linear orders on V, in which each element of S is greater than each element of $\overline{S}$. For any of these orders it is impossible to obtain any information regarding the relation between two elements of S or two elements of $\overline{S}$ using comparisons between an elements of S and an element of $\overline{S}$. Therefore, aside from these $n - x$ comparisons, there must be at least $c(k-1,x,m_1)$ comparisons to almost sort S and at least $c(k,n-x,m_2)$ comparisons to $\overline{S}$, where m_1, $m_2 \geq 0$ are integers satisfying $m_1 + m_2 \leq m$. This implies the following recursive inequality:

$$c(k,n,m) \geq c(k,n-x,m_1) + n - x + c(k-1,x,m_2)$$

where $m_1 + m_2 \leq m$.

By the inductive assumption:

$$c(k,n,m) > k\left(\frac{(n-x)^{1+1/k}}{d(1+m_1)^{1/(2k)}} - (n-x)\right) + (n-x) + (k-1)\left(\frac{x^{1+1/(k-1)}}{d(1+m_2)^{1/(2(k-1))}} - x\right)$$

By opening parentheses and permuting terms we get

$$c(k,n,m) > \frac{k}{d}\frac{(n-x)^{1+1/k}}{d(1+m_1)^{1/(2k)}} + \frac{k-1}{d}\frac{x^{1+1/(k-1)}}{(1+m_2)^{1/(2(k-1))}} + n - kn =$$

$$= \frac{k}{d}n^{1+1/k}\left[\frac{(1-\alpha)^{1+1/k}}{(1+m_1)^{1/(2k)}} + \left(1-\frac{1}{k}\right)\frac{\alpha^{1+1/(k-1)}\cdot n^{1/(k(k-1))}}{(1+m_2)^{1/(2(k-1))}} + \frac{1}{k}\cdot\frac{d}{n^{1/k}}\right] - kn$$

where $\alpha = x/n$.

Recall the geometric mean inequality: $\beta b + \gamma c \geq b^{\beta}c^{\gamma}$ where $\beta + \gamma = 1$, β, γ, b, $c \geq 0$. Applying it we conclude:

$$c(k,n,m) > \frac{k}{d}n^{1+1/k}\left[\frac{(1-\alpha)^{1+1/k}}{(1+m_1)^{1/(2k)}} + \frac{\alpha\cdot n^{1/k^2}}{(1+m_2)^{1/(2k)}}\cdot\frac{d^{1/k}}{n^{1/k^2}}\right] - kn.$$

Since $\left(1+\frac{1}{k}\right)^k < e < d$, $d^{1/k} > 1 + 1/k$, then:

$$c(k,n,m) > \frac{k}{d}\, n^{1+1/k}\left[\frac{(1-\alpha)^{1+1/k}}{(1+m_1)^{1/(2k)}} + \frac{\alpha(1+1/k)}{(1+m_2)^{1/(2k)}}\right] - kn.$$

But $m_1 + m_2 \le m$ so $m_1 \le m$ and $m_2 \le m$. Hence

$$c(k,n,m) > \frac{k}{d}\, n^{1+1/k}\left[\frac{(1-\alpha)^{1+1/k}}{(1+m)^{1/(2k)}} + \frac{\alpha(1+1/k)}{(1+m)^{1/(2k)}}\right] - kn.$$

Recall Bernoulli's inequality: $(1-\alpha)^t \ge 1 - \alpha t$ for $t \ge 1$, $\alpha \le 1$. This implies

$$c(k,n,m) > \frac{k}{d}\, n^{1+1/k}\left[\frac{1-\alpha(1+1/k)}{(1+m)^{1/(2k)}} + \frac{\alpha(1+1/k)}{(1+m)^{1/(2k)}}\right] - kn =$$
$$= \frac{k}{d}\,\frac{n^{1+1/k}}{(1+m)^{1/(2k)}} - kn = k\left(\frac{n^{1+1/k}}{d(1+m)^{1/(2k)}} - n\right).$$

This completes the proof of the inductive step.

The inductive proof must stop at one of the following base cases:

a) $n = 1$, $k \ge 1$ (and necessarily, $m = 0$).
In this case $k\left(\frac{n^{1+1/k}}{d(1+m)^{1/(2k)}} - n\right) < 0$ and the theorem holds trivially.

b) $k = 1$, $m \le \binom{n}{2}$. We have to prove that $c(1,n,m) > n^2/(d\sqrt{1+m}) - n$, or in other words: an algorithm that uses $p = n^2/(d\sqrt{1+m}) - n$ comparisons in one round leaves more than m unknown relation (in the worst case). This is proved using the methods for sorting in one round (see [16] and also [1]). From the graph representing the comparisons omit a set V_0 consisting of the $n/2$ vertices of the highest degree. The remaining part contains at least $n/2$ vertices and the highest degree is smaller than $4p/n \le 4n/(d\sqrt{1+m}) - 1$. The remaining graph can be partitioned into $t \le 4n/(d\sqrt{1+m})$ color classes $V_1, \ldots, V_t$, $|V_i| = x_i$, $i = 1, \ldots, t$.

We restrict the order to the case that an element from V_i is greater than an elements of V_j iff $i > j$. By convexity of the function z^2, the number of the unknown relations if at least

$$\sum_{i=1}^{t}\binom{x_i}{2} \ge t\binom{\frac{n/2}{t}}{2} = \frac{1}{2}t\frac{n}{2t}\left(\frac{n}{2t}-1\right) = \frac{n}{4}\left(\frac{n}{2t}-1\right) = \frac{n}{4}\left(\frac{d\sqrt{1+m}}{8}-1\right).$$

Thus, it is enough to prove that

$$\frac{n}{4}\left(\frac{d}{8}\sqrt{1+m}-1\right) > m.$$

One can easily check that it suffices to prove the last inequality for $m = \binom{n}{2}$. For this case we have to show that

$$\frac{n}{4}\left(\frac{d}{8}\sqrt{1+\frac{n(n-1)}{2}}-1\right) > \frac{1}{2}\,n(n-1)$$

or

$$\frac{d}{8}\sqrt{\frac{n^2-n+2}{2}} > 2n-1$$

but

$$\frac{d}{8}\sqrt{\frac{n^2-n+2}{2}} = \frac{d}{8\sqrt{2}}\sqrt{n^2-n+2} > \frac{d}{8\sqrt{2}}\,(n-1/2) = 2n-1.$$

This completes the proof of the theorem. ∎

Corollary 2.9. *(i) For* $p \geq 2n$ $\quad a(n,p,\varepsilon) = \Omega\left(\dfrac{\log(1/\varepsilon)}{\log(p/n)}\right)$,

(ii) for $p \leq 2n$ $\quad a(n,p,\varepsilon) = \Omega\left(\dfrac{\log(1/\varepsilon)}{p/n}\right)$.

Proof. (i) Using the last theorem $pk \geq k\left(\dfrac{n^{1+1/k}}{d(1+m)^{1/(2k)}} - n\right)$ so $1 + p/n \geq \dfrac{n^{1/k}}{d(1+m)^{1/(2k)}}$.

Hence for $m = \varepsilon n^2$

$$a(n,p,\varepsilon) \geq k \geq \frac{(1/2)\log(1/\varepsilon) - O(1)}{\log(1+p/n)} = \Omega\left(\frac{\log(1/\varepsilon)}{\log(p/n)}\right),$$

which proves (i).

(ii) Is a trivial consequence of 2.7. ∎

Proof of the lower bound of theorem 1.2.

Immediate from the last corollary and Theorem 2.5. ∎

Lemma 2.10. $m(n,p,\varepsilon) = \Omega\left(\dfrac{1}{\varepsilon p}\right)$ *for* $\varepsilon \geq 1/n$.

Proof. It suffices to prove a serial lower bound of $\Omega(1/\varepsilon)$. Clearly we may assume, say $\varepsilon < 1/10$. Partition each of the two sorted lists A and B into $t = \lfloor n/m \rfloor$ blocks of size at least $m = \lceil 4\varepsilon n \rceil$ consecutive elements each. Denote these blocks by A_i, B_i, $i = 1, \ldots, t$. We restrict ourselves to orders such that each elements of $A_i \cup B_i$ is smaller than each element of $A_j \cup B_j$ if $i < j$. Therefore, if less than $t/2 = \Omega(1/\varepsilon)$ comparisons were made, then there are at least $t/2$ pairs of A_i, B_i each that no comparisons were made between any element of A_i to any element of B_i and we have no information about their order relations. Therefore, the number of unknown order relations between elements is at least $(t/2) \cdot m^2 \geq n/(4m) \cdot m^2 = nm/4 \geq \varepsilon^2 n$ as needed. ∎

Lemma 2.11. $m(n,p,\varepsilon) = \Omega\left(\log \frac{\log 1/\varepsilon}{\log \varepsilon p}\right)$ *for* $p \geq 4/\varepsilon$, $\varepsilon \geq 1/n$.

Proof. The proof is similar to that of [20]. Let $\alpha = p/n$. Define $n^{(0)} = m$, $p^{(0)} = p$, $n^{(k+1)} = \left\lceil \frac{n^{(k)}}{8\sqrt{p^{(k)}}} \right\rceil$, $p^{(k)} = \alpha \cdot 8^k \cdot n^{(k)}$. We prove the following proposition by induction.

Proposition. *For* $k \leq (1/2)\log \frac{\log p}{6 \log \varepsilon p}$, *after* k *rounds it is possible that there are* $t^{(k)} = \left\lceil \frac{n}{8^k n^{(k)}} \right\rceil$ *sets of pairs* A_i, B_i, $i = 1, \ldots, t$ *where* A_i *consists of* $n^{(k)}$ *elements from the first list and* B_i *consists of* $n^{(k)}$ *elements of the second list, such that each elements of* $A_i \cup B_i$ *is smaller than each element of* $A_j \cup B_j$ *iff* $i < j$, *and all the merged orders of* A_i *and* B_i *are possible.*

For $k = 0$ the proposition is trivially true. Assume it is true for k, we prove it for $k+1$. Let E be the set of comparisons made at the $k+1$ round. There are at least $t^{(k)}/2$ pairs (A_i, B_i) such that no more than $|E_i| = 2p/t^{(k)} \leq 2p \cdot 8^k n^{(k)}/n = 2\alpha \cdot 8^k \cdot n^{(k)} \leq 2 \cdot p^{(k)}$ comparisons were made between them. We divide each such pair of lists into $\left\lfloor \frac{n^{(k)}}{n^{(k+1)}} \right\rfloor$ subsets of pairs of size at least $n^{(k+1)}$ each. Note that $\left\lfloor \frac{n^{(k)}}{n^{(k+1)}} \right\rfloor \geq \lfloor 8\sqrt{p^{(k)}}\left(1 - \frac{1}{n^{(k+1)}}\right) \rfloor \geq 6\lceil \sqrt{p^k} \rceil - 1$. Denote them as A_{ij}, B_{ij}, $j = 1, \ldots, 6\lceil \sqrt{p^{(k)}} \rceil - 1$. Let $E_{i,r,s}$ be the set of comparisons between A_{ir} and B_{is} and $E_{i\ell} = \bigcup_{r=1}^{3\lceil \sqrt{p^{(k)}} \rceil} E_{i,r,r+\ell}$ $0 \leq \ell \leq 3\lceil \sqrt{p^{(k)}} \rceil - 1$. Clearly $\sum_{\ell=0}^{3\lceil \sqrt{p^{(k)}} \rceil - 1} |E_{i\ell}| \leq |E_i| \leq 2 \cdot p^{(k)}$, therefore there exists an ℓ such that $|E_{i\ell}| \leq \frac{2p^{(k)}}{3\lceil \sqrt{p^{(k)}} \rceil} \leq (2/3)\sqrt{p^{(k)}}$. We restrict ourselves to orders such that any element of $A_{i,r} \cup B_{i,r+\ell}$ is smaller than $A_{i,s} \cup B_{i,s+\ell}$ if $r < s$ $(1 \leq r, s \leq 3\lceil \sqrt{p^{(k)}} \rceil)$. But $E_{i,\ell} = \bigcup_{r=1}^{3\lceil \sqrt{p^{(k)}} \rceil} E_{i,r,r+\ell}$, therefore, there are at least $2\lceil \sqrt{p^{(k)}} \rceil$ different values of r such that there is no information between $A_{i,r}$ and $B_{i,r+\ell}$. Thus there are $(t^{(k)}/2) \cdot 2 \cdot \lceil \sqrt{p^{(k)}} \rceil$ sets of pairs of size $n^{(k+1)}$. But

$$t^{(k)} \left\lceil \sqrt{p^{(k)}} \right\rceil \geq \frac{n}{8^k n^{(k)}} \cdot \sqrt{p^{(k)}} = \frac{n}{8^{k+1}} \Big/ \frac{n^{(k)}}{8\sqrt{p^{(k)}}} \geq \frac{n}{8^{k+1} n^{(k+1)}}$$

and because the left hand side is an integer we really have at least $t^{(k+1)}$ sets as needed to complete the proof of the proposition.

We complete the proof using the previous proposition.

First, we prove that $n^{(k)} \geq \frac{p^{1/2^k}}{8^k \cdot \alpha}$. For $k = 0$ it is clear. Assume it for k, we prove it for $k+1$

$$n^{(k+1)} \geq \frac{n^{(k)}}{8\sqrt{p^{(k)}}} = \frac{\sqrt{n^{(k)}}}{8\sqrt{\alpha \cdot 8^k}} \geq \frac{p^{1/2^{k+1}}}{8\sqrt{\alpha \cdot 8^k} \cdot \sqrt{\alpha \cdot 8^k}} = \frac{p^{1/2^{k+1}}}{8^{k+1} \cdot \alpha}$$

Next we note that unless $t^{(k)}(n^{(k)})^2 \leq \varepsilon n^2$ the algorithm cannot stop, because the order relations between the pairs are unknown. But

$$\varepsilon n^2 \geq t^{(k)}(n^{(k)})^2 \geq \frac{n}{8^k n^{(k)}} \cdot (n^{(k)})^2 = \frac{n}{8^k} \cdot n^{(k)} \geq \frac{n}{8^k} \cdot \frac{p^{1/2^k}}{8^k \cdot \alpha} = \frac{n^2 p^{1/2^k}}{p \cdot 2^{6k}}$$

means

$$\frac{1}{2^k} \log p \leq \log(\varepsilon p) + 6k$$

or

$$\log p \leq 2^k(6k + \log(\varepsilon p)) \leq 2^k \cdot 6k \cdot \log(\varepsilon p) \leq 6 \cdot 4^k \log(\varepsilon p)$$

so

$$k \geq \frac{1}{2} \log \frac{\log p}{6 \log \varepsilon p} \geq \frac{1}{2} \log \frac{\log 1/\varepsilon}{6 \log \varepsilon p} = \Omega\left(\log \frac{\log 1/\varepsilon}{\log \varepsilon p}\right) .$$

This completes the proof. ∎

Proof of the lower bound of Theorem 1.3.

Assume, first, that $1/2 \geq \varepsilon \geq 1/n$. For $p \geq 4/\varepsilon$ this follows from Lemma 2.11. For $p \leq 4/\varepsilon$ it follows from Lemma 2.10 and the lower bound in lemma 2.11 for $p = 4/\varepsilon$. If $\varepsilon \leq 1/n$ there is nothing to prove because even the lower bound for $\varepsilon = 1/n$ suffices. ∎

3. The Upper Bounds

In this section we prove the upper bounds in the theorems appearing in section 1. The section is organized as follows; we start with the rather easy proof of the upper bound for approximate merging. Then we consider a stronger definition of approximate sorting and establish some basic lemmas. This enables us to prove the upper bound for approximate sorting for the case $p \geq 2n$ (3.1–3.6). Next, we obtain the bound for $p \leq n/\log^* n$ and for $n/\log^* n \leq p \leq 2n$ (3.7–3.8) and hence complete the proof of the upper bound for approximate sorting. Finally approximate maximum is considered. A modification of (3.6) supplies the upper bound for $p \geq 2n$ (3.9), which is then used to obtain the bound for $p \leq 2n$ (3.10).

Remark. Throughout this section, we assume whenever it is needed, that n is sufficiently large.

Proof of the upper bound in Theorem 1.3.

Assume first that $\varepsilon \geq 4/n$. Take $t = \lfloor 4/\varepsilon \rfloor$ elements from each list such that the difference between the ranks of consecutive elements in each list is at most $\varepsilon n/3$ and each list contains a member in the top $\varepsilon n/3$ and the bottom $\varepsilon n/3$ elements of the corresponding set. Now merge these lists using Valiant's algorithm [30] (see also [25]). This costs $O\left(\frac{t}{p} + \log \frac{\log t}{\log(2+p/t)}\right) = O\left(\frac{1}{\varepsilon p} + \log \frac{\log 1/\varepsilon}{\log(2+\varepsilon p)}\right)$. One can easily check that the total number of unknown relations left between pairs of elements is at most $(2t+1)(\varepsilon n/3)^2 \leq \varepsilon n^2$ as needed.

For $\varepsilon \leq 4/n$ we simply perform a full merging. This completes the proof. ∎

We next discuss the upper bound for approximate sorting. First, notice that for $\varepsilon \leq 100/n$ we can simply apply full sorting which costs $O(\log n/\log(1+p/n))$ time (see [9], [13], [5]), as needed. Hence, we may assume that $\varepsilon \geq 100/n$. For $\varepsilon \geq 1/100$ we apply the same algorithm as for $\varepsilon = 1/100$. It thus suffices to consider the case $100/n \leq \varepsilon \leq 1/100$. It is convenient to introduce the following stronger definition of approximate sorting:

Definition. An ε-approximate sorting of the list N of cardinality $|N| = n$ is a sequence of elements of N : $\{x_i : i = 1, \ldots, \lfloor 1/\varepsilon \rfloor\}$ such that there are sets Sx_i and Bx_i satisfying $|Sx_i| > \varepsilon n(i - 1/5) - 1$, $|Bx_i| \geq n - \varepsilon n(i + 1/5) - 1$ and x_i is known to be bigger than each member of Sx_i and smaller than each member of Bx_i. In particular, $\varepsilon n(i-1/5) < r_N(x_i) < \varepsilon n(i+1/5)$ where $r_N(y)$ is the rank of the element y in the list N. Define $x_0 = -\infty$, $x_{\lfloor 1/\varepsilon \rfloor + 1} = +\infty$, $N_i = \{y \mid y$ is not known to be $< x_i$ or $> x_{i+1}\}$, $i = 0, \ldots, \lfloor 1/\varepsilon \rfloor$. Clearly, if we have an ε-approximate-sorting of N, then $|N_i| \leq (7/5)\varepsilon n + 1$ for each i and hence the total number of unknown order relations is at most:

$$\sum_{i=0}^{\lfloor 1/\varepsilon \rfloor} \binom{|N_i|}{2} \leq \left(\frac{1}{\varepsilon} + 1\right) \frac{1}{2} \left(\frac{7}{5}\varepsilon n + 1\right) \frac{7}{5}\varepsilon n \leq$$

$$\frac{7}{10}\varepsilon n^2 \left(\frac{7}{5} + \frac{1}{\varepsilon n} + \frac{7}{5}\varepsilon + \frac{1}{n}\right) \leq \frac{7}{10}\varepsilon n^2 \cdot \frac{10}{7} = \varepsilon n^2 .$$

Hence, an ε-approximate sorting supplies all order relations but at most εn^2. Recall that an ε-approximate maximum is an element x such that $r_N(x) \geq (1-\varepsilon)n$, i.e., an x and a subset $S \subset N$ such that x is known to be bigger than each element of S and $|S| \geq (1-\varepsilon)n$.

Lemma 3.1. [27]: *For every m and a, there is a graph with m vertices and at most $(2m^2 \log m)/a$ edges in which any two disjoint sets of $a+1$ vertices are joined by an edge.* ∎

Lemma 3.2. [27]: *If m elements are compared according to the edges of a graph in which any two disjoint sets of $a+1$ vertices are joined by an edge, then for every rank all but at most $6a \log m + a$ elements from the ones with a smaller rank will be known to be too small to have that rank. A symmetric statement holds for the elements with a bigger rank.* ∎

Proposition 3.3. *Assume we have m elements and $p = 2m^2 \log m/a$. Then one can find in one round (using p comparisons) a $35(a/m)\log m$-approximate sorting and a $(7a/m)\log m$ approximate maximum.*

Proof. Compare the elements according to the edges of the graph supplied by Lemma 3.1. For each admissible i, let A_i be the set of all elements y whose ranks $r(y)$ in the sorted list satisfy $\varepsilon m\,(i-1/5) < r(y) < \varepsilon m\,(i+1/5)$, where $\varepsilon = 35(a/m)\log m$. By applying Lemma 3.2 twice, we conclude (since $|A_i| > 2\cdot(6a(\log m+1))$, that at least one element $x_i \in A_i$ is known to satisfy $\varepsilon m\,(i-1/5) < r(x_i) < \varepsilon m\,(i+1/5)$. Taking Sx_i, Bx_i to be the sets of all elements known to be smaller (bigger, respectively) than x_i, we obtain the desired approximate sorting. The result for the maximum is similar. ∎

Proposition 3.4. *Suppose we have a set N partitioned into m pairwise disjoint sets N_i where $|N| = n$, and each $n_i = |N_i|$ is either $\lceil n/m\rceil$ or $\lfloor n/m\rfloor$. Suppose, further, that for each N_i we have an ε-approximate sorting. I.e., for $i = 1,\ldots,m$, $j = 1,\ldots,\lfloor 1/\varepsilon\rfloor$, $n_i\varepsilon\,(i-1/5) \le r_{N_i}(x_j^i) \le n_i\varepsilon\,(i+1/5)$. Put $x_0^i = -\infty$, $x^i_{\lfloor 1/\varepsilon\rfloor+1} = +\infty$. Let $X = \bigcup_i \bigcup_{j=i}^{\lceil 1/\varepsilon\rceil} x_j^i$, $|X| = t \le m/\varepsilon$ and assume we have a δ-approximate-sorting for X, i.e., $t\delta\,(i-1/5) \le r_X(y_i) \le t\delta\,(i+1/5)$ $i = 1,\ldots,\lfloor 1/\delta\rfloor$, then we have an $\varepsilon + 6\delta + 6m/n$ approximate-sorting for N.*

Proof. Let $S_k^i = \{\min j \mid x_j^i \text{ is known to be } \ge y_k\}$. By the definition of a δ-approximate sorting $r_X(y_k)$ is known to satisfy $r_X(y_k) \le \sum_{i=1}^{m} S_k^i \le t\delta\,(k+1/5)$. Similarly, $r_N(y_k)$ is known to satisfy

$$r_N(y_k) \le \sum_{i=1}^{m} \varepsilon n_i \left(S_k^i + 1/5\right) \le \varepsilon\cdot\frac{1}{5}\cdot n + \varepsilon\left(\frac{n}{m}+1\right)\delta\left(k+\frac{1}{5}\right)\frac{m}{\varepsilon} =$$

$$\frac{\varepsilon n}{5} + \delta n\left(1+\frac{m}{n}\right)\left(k+\frac{1}{5}\right).$$

Therefore (since $k \le 1/\delta$)

$$r_N(y_k) - \delta n k \le n\left(\frac{\varepsilon+\delta}{5} + \delta\frac{m}{n}\left(k+\frac{1}{5}\right)\right) \le$$

$$n\left(\frac{\varepsilon+\delta}{5} + \frac{m}{n}\left(1+\frac{\delta}{5}\right)\right) \le n\left(\frac{\varepsilon+\delta}{5} + \frac{6m}{5n}\right).$$

In the same way one can prove that

$$\delta n k - r_N(y_k) \le n\left(\frac{\varepsilon+\delta}{5} + \frac{6m}{5n}\right).$$

From the elements y_k it is easy to find a subset forming a $5\left(\delta + \frac{\varepsilon+\delta}{5} + \frac{6m}{5n}\right) =$ $\varepsilon + 6\delta + 6(m/n)$ approximate sorting (for the whole set N). ∎

Lemma 3.5. *Assume we have n elements, partitioned into $m \leq n/\log^6 n$ pairwise disjoint sets of size $n_i = \lfloor n/m \rfloor$ or $\lceil n/m \rceil$ each. Suppose that in each set we have an ε-approximate-sorting, where $\varepsilon \geq \dfrac{14}{(n/m)^{1/3}}$. Then one can find in one round using n processors an ε-approximate sorting of the total list of n elements where $\varepsilon' = \varepsilon + \dfrac{c}{(n/m)^{1/3}} \leq \varepsilon + \dfrac{c}{\log^2 n}$, where $c = 66$.*

Proof. Let X be the union of all the representing elements from all the sets, put $|X| = l \leq m/\varepsilon$. Define $a = \left\lceil \dfrac{2m^2 \log n}{\varepsilon^2 n} \right\rceil$. Clearly $\dfrac{2l^2 \log l}{a} \leq \dfrac{2(m/\varepsilon)^2 \log n}{2m^2 \log n/(\varepsilon^2 n)} = n$. Therefore we can use Proposition 3.3 to find a δ-approximate-sorting for the set X where

$$\delta = 35(a/l)\log l \leq 35\frac{2 \cdot 2m^2 \log n/(\varepsilon^2 n) \cdot \log n}{m/\varepsilon} \leq \frac{140 \log^2 n}{\varepsilon(n/m)}.$$

However, $\log^2 n \leq (n/m)^{1/3}$ and $(1/\varepsilon) \leq \dfrac{(n/m)^{1/3}}{14}$. Therefore

$$\delta \leq \frac{140(n/m)^{1/3}}{(n/m)} \frac{(n/m)^{1/3}}{14} = \frac{10}{(n/m)^{1/3}}.$$

By Proposition 3.4 one can find an ε'-approximate-sorting for

$$\varepsilon' = \varepsilon + \frac{60}{(n/m)^{1/3}} + \frac{6}{(n/m)} \leq \varepsilon + \frac{c}{(n/m)^{1/3}} \leq \varepsilon + \frac{c}{\log^2 n}. \qquad \blacksquare$$

Theorem 3.6. *For $p \geq 2n$, $a(n,p,\varepsilon) = O\left(\dfrac{\log(1/\varepsilon)}{\log \alpha} + \log^* n - \log^* \alpha\right)$ where $\alpha = p/n$.*

Proof. For each $1/4 \geq \varepsilon \geq 1/\alpha^{\log^* n - \log^* \alpha}$ we simply apply the algorithm described below for $\varepsilon = 1/\alpha^{\log^* n - \log^* \alpha}$, whose running time is $O(\log^* n - \log^* \alpha)$. Thus we can restrict ourselves to the case

$$\min\left(\frac{1}{100}, 1/\alpha^{\log^* n - \log^* \alpha}\right) \geq \varepsilon \geq \frac{100}{n}.$$

Let N be the given set of elements. We can assume that n is large enough (at each stage of the algorithm), otherwise exact sorting is done in a constant time. We consider two possible cases.

Case 1. $\varepsilon \leq \dfrac{2c}{\log^2 n}$. Partition N into $m = \dfrac{n}{c^7 \lfloor (1/\varepsilon)^6 \rfloor}$ sets each of size $n_i = \lfloor n/m \rfloor$ or $\lceil n/m \rceil$. $m \leq \dfrac{n}{c^7 \left(\log^2 n/(2c)\right)^6} \leq \dfrac{n}{\log^6 n}$, and $c^7 (1/\varepsilon)^6 \leq \lceil n/m \rceil \leq 2c^7 (1/\varepsilon)^6$.

Assign to each set of cardinality n_i, $n_i\alpha$ processors. Sort each of the sets using the algorithm in [5] (which is an acceleration of the AKS sorting network having $p_i \geq n_i$ processors). The complexity of this sorting is $O\left(\dfrac{\log(1/\varepsilon)^6}{\log(p_i/n_i)}\right) = O\left(\dfrac{\log(1/\varepsilon)}{\log \alpha}\right)$.

Take from each set an $\frac{c}{(n/m)^{1/3}}$ approximate sorting. Then use Lemma 3.5 to get $\frac{2c}{(n/m)^{1/3}}$ approximate sorting in one more round. But $\frac{2c}{(n/m)^{1/3}} \leq \frac{2c\varepsilon^2}{c^{7/3}} \leq \varepsilon$ as needed.

Case 2. $\varepsilon > \frac{2c}{\log^2 n}$. Partition N into $m = \lfloor n/\log^6 n \rfloor$ sets of size $n_i = \lfloor n/m \rfloor$ or $\lceil n/m \rceil$ each. For each such set assign $n_i \cdot \alpha$ processors. At each set, recursively, find an $\varepsilon - \frac{c}{\log^2 n}$ approximate sorting $\left(\varepsilon - \frac{c}{\log^2 n} > \frac{14}{(n/m)^{1/3}}\right)$ and use the previous lemma with one more round and only n processors to finish.

To complete the proof it suffices to establish the following facts;

(1) The algorithm can be at case 2 no more than $O(\log^* n - \log^* \alpha)$ before it arrives to case 1 with ε', p', n' (at each set) $\alpha = p'/n'$.

(2) $\varepsilon = O(\varepsilon')$ and therefore the complexity of the second case with the parameter ε' equals up to a constant factor to that for ε.

We first establish (2). It is clear that $\varepsilon \leq \varepsilon' + \sum_{i=1}^{k} \frac{c}{\log^2 n_i}$, where n_i is the sequence of the sizes of the sets of the different iterations. By the condition of case 2 $\varepsilon' > \frac{c}{\log^2 n_k}$. By the definition of n_i, $\frac{c}{\log^2 n_i} < (1/2)\frac{c}{\log^2 n_{i+1}}$ therefore $\varepsilon \leq \varepsilon' + \sum_{i=1}^{k} \frac{c}{\log^2 n_i} \leq \varepsilon' \left(1 + \sum_{i=0}^{k-1} \frac{1}{2^i}\right) < 3\varepsilon'$, i.e., $\varepsilon = O(\varepsilon')$.

Next we prove (1).

Let n_i be the maximal set of elements after i iteration at case 2. Therefore, $n_i \leq 2\log^6 n_{i-1}$ where $n_0 = n$. We prove, by induction on i, that $n_i < (8\log^{(i)} n)^6$.

Indeed, for $i = 0$ this is trivial and assuming it for i we have

$$n_{i+1} \leq 2\log^6 n_i \leq 2\log^6(8\log^{(i)} n)^6 \leq 2[6(3 + \log^{(i+1)} n)]^6 \leq (8\log^{(i+1)} n)^6 ,$$

as needed (in the last inequality we use the fact that n_i is still not small).

Therefore starting from n elements, case 2 continues at most as long as $\varepsilon' \geq \frac{2c}{\log^2 n_k}$. But $\varepsilon \geq \varepsilon' \geq \frac{2c}{\log^2 n_k} \geq \frac{2c}{n_k}$ or $n_k > (2c/\varepsilon)$. By the last claim this can happen at most $\log^* n - \log^*(1/\varepsilon) + O(1)$ times. But $1/\varepsilon \geq \alpha^{\log^* n - \log^* \alpha} \geq \alpha$ and therefore $\log^* n - \log^*(1/\varepsilon) + O(1) \leq \log^* n - \log^* \alpha + O(1)$ as needed. ∎

Lemma 3.7. *For each admissible ε and each $p \leq \frac{n}{\log^* n}$, $a(n,p,\varepsilon) = O\left(\frac{n\log(1/\varepsilon)}{p}\right)$.*

Proof. First, note that by the upper bound for $p \geq 2n$, (Theorem 3.6), for $p = n$, $O(\log 1/\varepsilon + \log^* n)$ rounds suffice. The algorithm for $p \leq n/\log^* n$ consists of iterations. We prove by induction that after the k'th iteration there are 2^k sets N_i, $i = 1, \ldots, 2^k$ $(\cup N_i = N)$ such that $n_i = |N_i| \leq q^k \cdot n$ for all i where $q = 7/8$, and each element of N_i is smaller than each element of N_j iff $i < j$. For $k = 0$, there is nothing to prove. Assuming the above for k, we prove it for $k + 1$. Define

$\alpha = n/p$. Assign to N_i, $p_i = \lceil n_i/(2\alpha)\rceil$ processors. Partition N_i into p_i sets each of size at most $\lceil (n_i/p_i)\rceil$ and using one processor for each such set find its median using the serial algorithm in $O(n_i/p_i) = O(\alpha)$ rounds. If $p_i > 1$, then for the set P_i of the p_i medians, we find a δ-approximate sorting with the p_i processors assigned to this set for $\delta = 1/4$ in $O(\log 1/\delta + \log^* p_i) = O(\log^* n)$ rounds. In this way we obtain for each i, an element x_i such that $\lfloor \frac{1}{3}p_i \rfloor \le r_{P_i}(x_i) \le \lfloor \frac{2}{3}p_i \rfloor$ and therefore $\left\lceil \frac{1}{8}n_i \right\rceil \le r_{N_i}(x_i) \le \left\lceil \frac{7}{8}n_i \right\rceil$. (In case $p_i = 1$ we obviously have such an element.) We next compare that element to all the other $n_i - 1$ elements in N_i in $O\left(\frac{n_i}{p_i}\right) = O(\alpha)$ rounds and split the set N_i into 2 sets N_{2i-1}, N_{2i} of size $\le \frac{7}{8}n_i$ each, such that each element of N_{2i-1} is smaller than each element of N_{2i}. This completes the iteration in $O(\alpha + \log^* n + \alpha) = O(\alpha)$ rounds ($\alpha \ge \log^* n$). Put $k = \frac{\log(8/\varepsilon)}{\log(8/7)}$. After the kth iteration $n_i \le \frac{\varepsilon}{8}n$, so it is easy to find $\lfloor 1/\varepsilon \rfloor$ representing elements y_i, such that $\varepsilon\left(i - \frac{1}{5}\right)n < r_N(y_i) < \varepsilon\left(i + \frac{1}{5}\right)n \quad (i = 1, \ldots, \frac{1}{\varepsilon})$ and the complexity of the algorithms is $O(\alpha \log 1/\varepsilon) = O\left(\frac{n}{p}\log 1/\varepsilon\right)$, as needed. ∎

Lemma 3.8. *For each admissible ε and $\frac{n}{\log^* n} \le p \le 2n$,*

$$a(n, p, \varepsilon) = O\left(\frac{n \log 1/\varepsilon}{p} + \log^* n\right).$$

Proof. If $\varepsilon \le 1/2^{\log^* n}$ then we simply simulate each round of the algorithm of $2n$ processors by $O\left(\frac{n}{p}\right)$ rounds with p processors. The time complexity is $O\left(\frac{n}{p}(\log 1/\varepsilon + \log^* n)\right) = O\left(\frac{n}{p}\log 1/\varepsilon\right)$ as needed. Thus we are left with the case $\varepsilon \ge 1/2^{\log^* n}$. Split the n elements in $m = \lfloor \varepsilon p/24 \rfloor$ sets of size $\lceil n/m \rceil$ or $\lfloor n/m \rfloor$ each. For each such set we assign p/m processors and find an ε'-approximate sorting where $\varepsilon' = \frac{\varepsilon}{4}$ in an algorithm which we describe below. This gives $\frac{4}{\varepsilon} \cdot \lfloor \frac{\varepsilon p}{24} \rfloor \le p$ elements. Since we have p processors we can, in $O(\log^* n)$ rounds, find an ε''-approximate sorting for these elements with $\varepsilon'' = \frac{1}{16 \cdot 2^{\log^* n}}$. From this one can apply Lemma 3.4 to produce a δ-approximate sorting for the original set of elements, where

$$\delta \le \frac{\varepsilon}{4} + \frac{6}{16 \cdot 2^{\log^* n}} + \frac{6m}{n} \le \frac{\varepsilon}{4} + \frac{6\varepsilon}{16} + \frac{6\varepsilon p}{24n} \le \varepsilon$$

as needed. It remains to explain the algorithm for $n' = \lceil \frac{n}{m} \rceil \le 48\frac{n}{p} \cdot \frac{1}{\varepsilon}$, $p' = \frac{24}{\varepsilon}$, $\varepsilon' = \frac{\varepsilon}{4}$.

If $p' \le \frac{n'}{\log^* n'}$, we are done (by Lemma 3.7) in $O\left(\frac{n'}{p'}\log(1/\varepsilon')\right) = O\left(\frac{n}{p}\log(1/\varepsilon)\right)$, as needed. Thus we may assume $p' \ge \frac{n'}{\log^* n'}$ and hence $\varepsilon' = \frac{1}{4} \cdot \varepsilon = \frac{1}{4} \cdot \frac{24}{p'} \le 6 \cdot \frac{\log^* n'}{n'} \le 1/2^{\log^* n'}$. But for $\varepsilon' \le 1/2^{\log^* n'}$ we are done again as in the beginning of the proof, by simulation of the algorithm with n' processors by p' ones, in time $O\left(\frac{n'}{p'}(\log(1/\varepsilon') + \log^* n')\right) = O\left(\frac{n}{p}\log(1/\varepsilon)\right)$. This completes the proof. ∎

Proof of the upper bound of Theorem 1.2.
Immediate from Lemmas 3.6, 3.7, 3.8. ∎

We conclude this section by considering the approximate maximum problem. (See also [3] for a more detailed account of a special case).

Theorem 3.9. *For each ε and $p \geq 2n$*

$$r(n,p,\varepsilon) = O\left(\log \frac{\log 1/\varepsilon}{\log \alpha} + \log^* n \cdot \log^* \alpha\right),$$

where $\alpha = p/n$.

Proof. For each $1/2 \geq \varepsilon \geq 1/\alpha^{2^{\log^* n - \log^* \alpha}}$ we simply apply the algorithm for $\varepsilon = 1/\alpha^{2^{\log^* n - \log^* \alpha}}$ described below. Therefore, we can restrict ourselves to the case $1/2^{2^{\log^* n - \log^* \alpha}} \geq \varepsilon \geq \frac{1}{n}$. The proof is very similar to that of the upper bound for approximate sorting and differs only in the following facts:

First we have an easy proposition that asserts that given a set N partitioned into m almost equal sets and given an ε-approximate maximum in each set and a δ-approximate maximum of these m representatives then we have an $\varepsilon + \delta + \frac{m}{n}$ approximate maximum in the whole set. Next, an analog of Lemma 3.5 obtained by replacing the word "sorting" by "maximum" can be proved (and in fact a stronger estimate holds).

The rest of the proof for $p \geq 2n$ is analogous to that of Theorem 3.6. The only essential change is the replacement of the AKS sorting network by Valiant's algorithm for the maximum problem ([30]). ∎

It remains to establish the upper bound for $p \leq 2n$.

Lemma 3.10. *For $p \leq 2n$ and all admissible ε, $r(n,p,\varepsilon) = O\left(\frac{n}{p} + \log\log\frac{1}{\varepsilon} + \log^* n\right)$.*

Proof. First we observe that the upper bounds for $p \geq 2n$ gives that for $p \geq n$, $O(\log\log 1/\varepsilon + \log^* n)$ rounds suffice.

Assuming we have $p \leq n$ processors, partition the n elements into p sets, each of size at most $\lceil n/p \rceil$. In each set find the maximum using one processor in at most $\frac{n}{p}$ rounds. Then from the p maximum elements we find an element in the top εp elements in $O(\log\log 1/\varepsilon + \log^* p) = O(\log\log 1/\varepsilon + \log^* n)$ rounds. (This can be done by the observation above.) Clearly, this element is in the top $\varepsilon p \cdot \frac{n}{p} = \varepsilon n$ elements as needed. The total number of rounds is thus $O\left(\frac{n}{p} + \log\log 1/\varepsilon + \log^* n\right)$.

Proof of the upper bound of Theorem 1.1.
This is a simple consequence of 3.9 and 3.10. ∎

We have thus completed the proofs of Theorems 1.1–1.3. As already mentioned, Proposition 1.0 (proved in [3]) is a special case of Theorem 1.1, with a somewhat better estimate. Its detailed proof appears in [3]. Note that the lower bound in this Proposition is a special case of Theorem 3.5.

4. Concluding Remarks and Open Problems

We have determined the exact behavior, up to a constant factor, of three of the main comparison problems; sorting, merging and selecting the maximum, even when we just want to have an ε-approximation for them. There is a fourth important comparison problem which is the general selection. We can define an ε-approximate selection for the rank βn, $\frac{1}{n} \leq \beta \leq 1$, $\frac{1}{2n} \leq \varepsilon \leq \frac{1}{2}$ as finding an element x in the set N whose rank is known to satisfy $n(\beta - \varepsilon) \leq r_N(x) \leq n(\beta + \varepsilon)$. Approximate maximum is thus the case where $\beta = 1$. Approximate median is the case $\beta = 1/2$. It is known that the algorithms for selection are harder than the algorithms for finding the maximum. However the complexity of parallel selection in the comparison models for every n and p is the same as for the maximum and is: $\Theta\left(\frac{n}{p} + \log \frac{\log n}{\log(2+p/n)}\right)$ (see [7], [27], [12]).

The exact complexity of approximate parallel selection is not known. Our lower bound for approximate maximum holds, of course, for approximate selection as well. On the other hand the upper bound for ε-approximate sorting gives an upper bound for approximate selection. In fact, our methods enable us to prove a slightly better upper bound that gives, for example, for $p = n$ the following upper bound.

$$O\left(\log\log 1/\varepsilon \frac{\log^* n - \log^* 1/\varepsilon + 2}{\log(\log^* n - \log^* 1/\varepsilon + 2)} + \log^* n\right) .$$

Note that this is really a better bound than the one for approximate sorting. In fact it is not more than $\log^* n / \log\log^* n$ times the approximate maximum lower bound. It is interesting to find the exact complexity of approximate selection and to decide whether it is more than the complexity for approximate maximum.

Acknowledgement. We would like to thank N. Pippenger for bringing some of the problems considered in this paper to our attention.

References

[1] N. ALON, and Y. AZAR: Sorting, approximate sorting and searching in rounds, *SIAM J. Discrete Math.* **1** (1988), 269–280.

[2] N. ALON, and Y. AZAR: The average complexity of deterministic and randomized parallel comparison sorting algorithms, Proc. 28th IEEE FOCS, Los Angeles, CA 1987, IEEE Press, 489–498; Also: *SIAM J. Comput.* **17** (1988), 1178–1192.

[3] N. ALON, and Y. AZAR: Finding an approximate maximum, *SIAM J. Comput.* **18** (1989), 258–267.

[4] N. ALON, and Y. AZAR: Parallel comparison algorithms for approximation problems, Proc 29th IEEE FOCS, Yorktown Heights, NY 1988, IEEE Press, 194–203.

[5] N. ALON, Y. AZAR, and U. VISHKIN: Tight complexity bounds for parallel comparison sorting, Proc. 27th IEEE FOCS, Toronto, 1986, 502–510.

[6] S. AKL: Parallel Sorting Algorithm, Academic Press, 1985.

[7] M. AJTAI, J. KOMLÓS, W.L. STEIGER, and E. SZEMERÉDI: Deterministic selection in $O(\log \log n)$ parallel time, Proc. 18th ACM STOC, Berkeley, California, 1986, 188–195.

[8] M. AJTAI, J. KOMLÓS, W.L. STEIGER, and E. SZEMERÉDI: Almost sorting in one round, Advances in Computing Research, to appear.

[9] M. AJTAI, J. KOMLÓS, and E. SZEMERÉDI: An $O(n \log n)$ sorting network, Proc. 15th ACM STOC (1983), 1–9; Also, M. AJTAI, J. KOMLÓS, and E. SZEMERÉDI: Sorting in $c \log n$ parallel steps, Combinatorica **3** (1983), 1–19.

[10] N. ALON: Expanders, sorting in rounds and superconcentrators of limited depth, Proc. 17th ACM STOC (1985), 98–102.

[11] N. ALON: Eigenvalues, geometric expanders, sorting in rounds and Ramsey Theory, *Combinatorica* **6** (1986), 207–219.

[12] Y. AZAR, and N. PIPPENGER: Parallel selection, *Discrete Applied Math.* **27** (1990), 49–58.

[13] Y. AZAR, and U. VISHKIN: Tight comparison bounds on the complexity of parallel sorting, *SIAM J. Comput.* **3** (1987), 458–464.

[14] B. BOLLOBÁS, and G. BRIGHTWELL: Graphs whose every transitive orientation contains almost every relation, *Israel J. Math.*, **59** (1987), 112–128.

[15] B. BOLLOBÁS: *Extremal Graph Theory*, Academic Press, London and New York, 1978.

[16] B. BOLLOBÁS, and M. ROSENFELD: Sorting in one round, *Israel J. Math.* **38** (1981), 154–160.

[17] B. BOLLOBÁS, and A. THOMASON: Parallel sorting, *Discrete Applied Math.* **6** (1983), 1–11.

[18] B. BOLLOBÁS, and P. HELL: Sorting and Graphs, in *Graphs and Orders*, I. Rival ed., D. Reidel (1985), 169–184.

[19] B. BOLLOBÁS: *Random Graphs*, Academic Press (1986), Chapter 15 (Sorting algorithms).

[20] A. BORODIN, and J.E. HOPCROFT: Routing, merging and sorting on parallel models of computation, *J. Comput. System Sci.* **30** (1985), 130–145. Also: Proc. 14th ACM STOC (1982), 338–344.

[21] R. HÄGGKVIST, and P. HELL: Graphs and parallel comparison algorithms, *Congr. Num.* **29** (1980), 497–509.

[22] R. HÄGGKVIST, and P. HELL: Parallel sorting with constant time for comparisons, *SIAM J. Comput.* **10** (1981), 465–472.

[23] R. HÄGGKVIST, and P. HELL: Sorting and merging in rounds, *SIAM J. Algeb. and Disc. Math.* **3** (1982), 465–473.

[24] D.E. KNUTH: *The Art of Computer Programming*, **3** *Sorting and Searching*, Addison Wesley 1973.

[25] C.P. KRUSKAL: Searching, merging and sorting in parallel computation, *IEEE Trans. Comput. 32* (1983), 942–946.

[26] F.T. LEIGHTON: Tight bounds on the complexity of parallel sorting, Proc. 16th ACM STOC (1984), 71–80.

[27] N. PIPPENGER: Sorting and selecting in rounds, *SIAM J. Comput.* **6** (1986), 1032–1038.

[28] S. SCHEELE: Final report to office of environmental education, Dept, of Health, Education and Welfare, Social Engineering Technology, Los Angeles, CA, 1977.

[29] Y. SHILOACH, and U. VISHKIN: Finding the maximum, merging and sorting in a parallel model of computation, *J. Algorithms* **2** (1981), 88–102.

[30] L.G. VALIANT: Parallelism in comparison problems, *SIAM J. Comp.* **4** (1975), 348–355.

N. Alon

The Raymond and Beverly Sackler
Faculty of Exact Sciences
Tel Aviv University
Tel Aviv, Israel
NOGA@MATH.TAU.AC.IL

Y. Azar

The Raymond and Beverly Sackler
Faculty of Exact Sciences
Tel Aviv University
Tel Aviv, Israel
AZAR@TAMUZ.STANFORD.EDU